มุม
ภาษา C

โปรแกรมเรสซิเดนต์ด้วยเทอร์โบซี

■ จเร เลิศสุดวิชัย

เทคนิคโปรแกรมในฉบับนี้ จะเป็นการเสนอวิธีการเขียนโปรแกรมเรสซิเดนต์ซึ่งเป็นตอนที่สอง ต่อจากฉบับ 50 โดยจะกล่าวถึงเทคนิคและวิธีการที่จะช่วยให้โปรแกรมเรสซิเดนต์ทำงานได้มากขึ้นกว่าเดิม เช่น การกำหนดแป้นพิมพ์เพื่อปลุกโปรแกรมเรสซิเดนต์ให้ตื่นขึ้นมาทำงาน การเปิดไฟล์สำหรับการอ่านหรือเขียน การติดต่อหรือส่งข้อมูลให้กับโปรแกรมเรสซิเดนต์ ซึ่งได้ฝังตัวอยู่ในหน่วยความจำก่อนแล้ว และการกำจัดหรือถอนโปรแกรมเรสซิเดนต์ออกจากหน่วยความจำ ซึ่งวิธีการทั้งหมดจะต้องอาศัยความเข้าใจการทำงานของดอสและไบออสในบางส่วน ฟังก์ชันบริการของดอสที่ไม่เปิดเผย (Undocument) บางฟังก์ชันที่ต้องนำมาใช้งานก็จะนำมาอธิบายวิธีการเรียกใช้ โดยจะอธิบายแยกออกเป็นส่วน ๆ ซึ่งผู้อ่านสามารถนำไปใช้เขียนโปรแกรมด้วยภาษาอื่น เช่น แอสเซมบลี หรือปาสคาล แต่ตัวอย่างโปรแกรมที่ใช้ในบทความฉบับนี้ก็จะเป็นเช่นดังเคย คือจะมีทั้งโปรแกรมที่เขียนด้วยเทอร์โบซีและไมโครซอฟต์ซี ซึ่งสุดท้ายจะแยกเขียนเป็นโมดูล ผู้อ่านสามารถสร้างหรือเขียนโปรแกรมเรสซิเดนต์ที่ทำงานตามต้องการ แล้วนำมาลิงค์รวมกันโดยไม่ต้องมาจัดการเรื่องการติดตั้งโปรแกรมเรสซิเดนต์เลย

การคำนวณหาขนาดของโปรแกรม

การติดตั้งโปรแกรมเรสซิเดนต์ได้กล่าวไปบ้างแล้วในตอนที่หนึ่ง การคำนวณขนาดโปรแกรมที่จะฝังในหน่วยความจำ ซึ่งขนาดของสแต็กต้องกำหนดตามความเหมาะสมของโปรแกรม และขนาดของโปรแกรมก็ขึ้นกับโมเดลของโปรแกรมที่เขียนด้วยเช่นกัน การอ่านค่าและการเปลี่ยนค่าในตารางอินเตอร์รัพต์เวกเตอร์ จะไม่กล่าวถึงในที่นี้อีก

การปลุกโปรแกรมเรสซิเดนต์

โปรแกรมเรสซิเดนต์ส่วนใหญ่แล้วมักจะปลุกหรือเรียกโปรแกรมรูทีนที่จะป็อบอัพให้ขึ้นมาทำงานโดยการกดคีย์พิเศษ (Hot Key) ที่กำหนดเอาไว้ การทำงานลักษณะนี้ทำได้โดยการเขียนโปรแกรมอินเตอร์รัพต์ที่เกี่ยวกับคีย์บอร์ดใหม่ เมื่อมีการกดคีย์พิเศษโปรแกรมอินเตอร์รัพต์คีย์บอร์ดใหม่ที่ถูกเขียนขึ้นก็จะทำการเรียกไปยังรูทีนหรือฟังก์ชันที่ต้องการให้ทำงาน อินเตอร์รัพต์ที่เกี่ยวกับคีย์บอร์ดที่ใช้กันอยู่มี 2 อินเตอร์รัพต์ คือ อินเตอร์รัพต์ 09h ซึ่งเป็นฮาร์ดแวร์อินเตอร์รัพต์ และอินเตอร์รัพต์ 16h นั้น เป็นซอฟต์แวร์อินเตอร์รัพต์ (เป็นบริการของไบออสในการอ่านคีย์) ความแตกต่างระหว่างทั้งสองอินเตอร์รัพต์คือ อินเตอร์รัพต์ 16h นั้นเป็นซอฟต์แวร์อินเตอร์รัพต์ จะเกิดขึ้นก็ต่อเมื่อมีการเรียกใช้จากโปรแกรมอื่น ๆ (อินเตอร์รัพต์ 16h มีฟังก์ชันบริการอยู่หลายอย่างที่สำคัญ คือ อ่านตัวอักษรถัดไปจากบัฟเฟอร์ของคีย์บอร์ด ตรวจสอบว่ามีตัวอักษรพร้อมจะนำเข้าหรือไม่ และรายงานสถานภาพของคีย์บอร์ด ซึ่งได้แก่ ปุ่ม Shift, Caps-lock, Num-lock, Scroll-lock, Insert, Ctrl และ Alt) ส่วนอินเตอร์รัพต์ 9h นั้น เป็นฮาร์ดแวร์อินเตอร์รัพต์ ซึ่งจะเกิดขึ้นทุกครั้งที่มีการกดหรือปล่อยคีย์บอร์ด การทำงานของอินเตอร์รัพต์นี้จะอ่านค่ารหัสสแกน (scan code) ของคีย์ที่ถูกกดจากพอร์ต 60h (อ่านมา 1 ไบต์) แล้วทำการเปลี่ยนเป็นรหัสแอสกี (เปลี่ยนเป็น 2 ไบต์ คือ ไบต์ต่ำ เป็นรหัสแอสกีของคีย์ที่กดและไบต์สูงเป็นรหัสสแกน) แล้วนำไปเก็บไว้นำบัฟเฟอร์ของคีย์บอร์ด สำหรับใช้งานโดยโปรแกรมอื่น ๆ ซึ่งการอ่านก็จะผ่านทางอินเตอร์รัพต์หมายเลข 16h นั่นเอง

จากการทำงานของอินเตอร์รัพต์ทั้งสองที่กล่าวมาแล้ว จะพบว่าอินเตอร์รัพต์หมายเลข 09h จะตอบสนองทันทีทันใดได้ดีกว่าอินเตอร์รัพต์ 16h แต่ปัญหาที่สำคัญคือ ถ้าใช้อินเตอร์รัพต์หมายเลข 09h เป็นตัวปลุกรูทีนป็อบอัพ ในขณะที่โปรแกรมเรสซิเดนต์ตื่นขึ้นมาพร้อมที่จะทำงาน สภาวะนั้นดอสอาจจะทำงานอยู่ในฟังก์ชันที่เป็น Non-reentran ก็ได้ ซึ่งจะไม่ปลอดภัยสำหรับการทำงานของโปรแกรมหรือรูทีนป็อบอัพ แต่อย่างไรก็ตาม สามารถจะหลบเลี่ยงได้โดยการตรวจสอบสภาวะของดอสก่อนที่จะให้โปรแกรมหรือรูทีนป็อบอัพ ในตัวอย่างโปรแกรมจะใช้อินเตอร์รัพต์ 09h ในการปลุกโปรแกรมเรสซิเดนต์ โดยการเปลี่ยนแปลงการทำงานของอินเตอร์รัพต์ 09h ให้ตรวจสอบคีย์ที่ได้รับการกดกับคีย์พิเศษที่กำหนดขึ้นมา ถ้าตรงกันจะยังไม่เรียกโปรแกรมหรือรูทีนป็อบอัพทำงานทันที แต่จะทำการเซตค่าแฟล็กมาแล้วจะใช้อินเตอร์รัพต์ที่เกิดจากสัญญาณนาฬิกา (Time Interrupt) คืออินเตอร์รัพต์ 1ch และอินเตอร์รัพต์ 28h ตรวจสอบแฟล็กที่ถูกเซตค่านี้ พร้อมกับตรวจสอบสภาวะของดอสด้วย ซึ่งถ้าดอสอยู่ในสภาวะหรือทำงานฟังก์ชันที่ปลอดภัย (ปลอดภัยอย่างไรนั้น ต้องอ่านหัวข้อถัดไป)

จึงจะเรียกโปรแกรมหรือรูทีนป๊อบอัพขึ้นมาทำงาน

ในการกำหนดคีย์พิเศษนั้น สามารถกำหนดเป็นคีย์รวมระหว่างคีย์ Ctrl, Alt, Shift-left หรือ Shift-right กับคีย์อื่น ๆ ได้ ซึ่งคีย์ Ctrl, Alt, Shift-left และ Shift-right สามารถตรวจสอบได้จากตำแหน่งในพื้นที่ของไบออส 0 : 417 โดยค่าต่าง ๆ จะอยู่ในตารางที่ 1 (สาเหตุที่ไม่ใช้คีย์ Insert, Num lock, Scroll-lock และ Cap-lock ก็เพราะว่า สถานะของคีย์พวกนี้จะเปลี่ยนแปลงไปมาเมื่อกดคีย์แต่ละครั้ง แต่สำหรับคีย์ Ctrl, Alt, Shift-left และ Shift-right สถานะจะถูกเซตก็เมื่อได้รับการกดคีย์ค้างเอาไว้ และเมื่อปล่อยคีย์สถานะจะถูกเคลียร์ ซึ่งสามารถตรวจสอบได้ง่ายกว่า) และสำหรับคีย์อื่นนั้นจะถูกอ่านตรง ๆ จากพอร์ต 60h ค่าที่ได้จะเป็นรหัสสแกนดังในตารางที่ 2 ซึ่งแสดงรหัสสแกนประจำคีย์แต่ละคีย์

ข้อจำกัดของดอส

(Non reentrant)

เนื่องจากดอสถูกออกแบบม
โอเอสที่ทำงานแบบ Single task ดังน
ฟังก์ชันบริการของดอส (int 21h) จึงมีข้อจำกัด คือไม่สามารถจะเรียกใช้งานซ้ำซ้อนได้ (Non-reentrant) การเรียกใช้ฟังก์ชันของดอส ดอสจะเก็บค่าของสแต็กรีจิสเตอร์เอาไว้ แล้วเปลี่ยนมาใช้สแต็กของดอสซึ่งดอสสร้างขึ้นเอง พร้อมกับเก็บค่ารีจิสเตอร์ที่สำคัญลงสแต็กเมื่อทำงานเสร็จสิ้นจึงจะคืนค่าต่าง ๆ ของรีจิสเตอร์รวมทั้งสแต็ก
ที่
เขียนที่
ฟังก์ชันของดอ
จบฟังก์ชันดอสและออก
รูทีนป๊อบอัพแล้ว การทำงานของฟังก์ชันดอสที่เกิดจากการเรียกใช้ครั้งแรกก็จะไม่ถูกต้อง เนื่องจากค่าในสแต็กเดิมถูกทำลาย

ตารางที่ 1 สถานะของคีย์บอร์ดที่ตำแหน่งหน่วยความจำ 0000 : 0417h

บิตที่ 7	6	5	4	3	2	1	0	ความหมาย
*	.	.	.	.	.	.	.	Insert
.	*	.	.	.	.	.	.	Caps lock
.	.	*	.	.	.	.	.	Num lock
.	.	.	*	.	.	.	.	Scorll lock
.	.	.	.	*	.	.	.	Alt
.	.	.	.	.	*	.	.	Ctrl
.	.	.	.	.	.	*	.	Left Shift
.	.	.	.	.	.	.	*	Right Shift

Active = 1
Non Active = 0

ตารางที่ 2
รหัสสแกนประจำคีย์บอร์ดแต่ละคีย์

คีย์ฟังก์ชัน		คีย์ที่เป็นตัวอักษร								คีย์ตัวเลข	
ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส	ชื่อคีย์	รหัส
F1	59	'	41	Tab	15	A	30	Z	44	ESC	1
F2	60	1	2	Q	16	S	31	X	45	Home	71
F3	61	2	3	W	17	D	32	C	46	Up	72
F4	62	3	4	E	18	F	33	V	47	PgUp	73
F5	63	4	5	R	19	G	34	B	48	PrtSc	55
F6	64	5	6	T	20	H	35	N	49	←	75
F7	65	6	7	Y	21	J	36	M	50	5	76
F8	66	7	8	U	22	K	37	,	51	→	77
F9	67	8	9	I	23	L	38	.	52	Minus	74
F10	68	9	10	O	24	;	39	/	53	End	79
		0	11	P	25	'	40	Right Shift	54	Down	80
		-	12	[	26	left Shift	42	ALT	56	PgDn	81
		=	13	]	27			Spacbar	57	Ins	82
		\	43	Enter	28			Caps look	58	Del	83
		back space	14							Plus	78

ได้รับการกดคีย์พิเศษเพื่อปลุก
เรสซิเดนต์ขึ้นมา ดังที่กล่าวใน
ที่แล้วในอินเตอร์รัพต์ของคีย์บอร์ด
นเตอร์รัพต์ 09h) จะเซตค่าแฟล็ก pop-
flag ให้มีค่าเป็น 1 เพื่อเตรียมพร้อมสำหรับ
การเรียกโปรแกรมหรือรูทีนป๊อบอัพขึ้นมา
ทำงาน จากนั้นอินเตอร์รัพต์ไทเมอร์ (1ch)
ซึ่งกระตุ้น 18.2 ครั้งต่อนาที จะคอยตรวจ-
สอบ popflag และ InDOS (มีค่าเป็น 0)
ถ้าพร้อมก็จะเรียกโปรแกรมหรือรูทีนป๊อบ-
อัพขึ้นมาทำงานทันที (ความจริงยังไม่
สามารถเรียกโปรแกรมหรือรูทีนป๊อบอัพ
ขึ้นมาทำงานได้ทันที แต่จะต้องตรวจสอบ
สภาพของการทำงานดิสค์ก่อน ซึ่งจะ
กล่าวในหัวข้อถัดไป)

มีกรณีพิเศษอยู่กรณีหนึ่งที่โปรแกรม
หรือรูทีนป๊อบอัพสามารถทำงานได้ทันที
เมื่อค่าใน InDOS เป็น 1 คือในขณะที่ดอส
รอการกดคีย์ (ขณะอยู่ในเครื่องหมายเตรียม
พร้อม) ดอสจะเรียก อินเตอร์รัพต์ 028h
ซึ่งคือสถานะที่ดอสว่างอยู่ โปรแกรมหรือ
รูทีนป๊อบอัพสามารถทำงานได้โดยไม่ต้อง
ตรวจสอบเงื่อนไขอื่นอีก

โปรแกรมเรสซิเดนต์กับการทำงานของดิสค์

ควรหลีกเลี่ยงการปลุกโปรแกรม
หรือรูทีนป๊อบอัพขึ้นมาทำงานระหว่างอยู่
ในการทำงานของดิสค์ เพราะการควบคุมดิสค์
การอ่าน การเขียน และการค้นหา (seek)
ข้อมูลในดิสค์ต้องอาศัยช่วงเวลา (Timing)
ที่แน่นอน การขัดจังหวะกลางคันของโปร-
แกรมเรสซิเดนต์ อาจจะทำความเสียหาย
ให้กับดิสค์

ดังนั้นเพื่อแก้ไขปัญหานี้โปรแกรม-
เรสซิเดนต์จะทำการเปลี่ยนแปลงการทำงาน
ของอินเตอร์รัพต์บริการเรื่องดิสค์คือ ไบออส
อินเตอร์รัพต์ 13h โดยจะกำหนดแฟล็กขึ้น
แกรมเรสซิเดนต์ทำงาน (ป๊อบอัพ) ซึ่งจะ
ตรวจสอบเฉพาะในอินเตอร์รัพต์ 1ch เท่านั้น
(ในอินเตอร์รัพต์ 28h ไม่ต้องตรวจสอบ
แฟล็กอันนี้) ซึ่งแฟล็กที่กำหนดจะถูกเซต
ค่าให้เป็น 1 เมื่อเข้าไปอินเตอร์รัพต์ 13h
และเมื่อออกจากอินเตอร์รัพต์ 13h ก็จะ
เคลียร์ค่าแฟล็กให้เป็น 0

การทำงานภายในรูทีนป๊อบอัพ

หลังจากสภาวะต่าง ๆ รวมทั้งดอส
พร้อมที่จะใช้รูทีนป๊อปอัพขึ้นมาทำงาน
โปรแกรมเรสซิเดนต์จะต้องทำการเก็บค่า
ต่าง ๆ ของสิ่งแวดล้อมที่เป็นอยู่เอาไว้
ก่อนเรียกรูทีนป๊อบอัพ และเมื่อทำงานใน
รูทีนป๊อบอัพเสร็จแล้วก็ต้องคืนค่าต่าง ๆ ที่
เก็บไว้ให้ถูกต้องด้วย ซึ่งงานที่เกี่ยวข้อง
ต้องทำนั้นมีดังนี้

- สแต็ก
- อินเตอร์รัพต์ของ Control-Break และ Control-C
- อินเตอร์รัพต์ของ Critical error
- Program Segment prefix
- พื้นที่รับส่งข้อมูลกับดิสค์ (Disk transfer address, DTA)
- จอภาพและเคอร์เซอร์

โดยจะขออธิบายรายละเอียดแยกเป็นหัวข้อต่อไปนี้

การเปลี่ยนแปลงสแต็ก

ในขณะที่โปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาทำงาน สแต็กที่ใช้งานอยู่จะเป็นสแต็กของโปรแกรมที่ทำงานค้างอยู่หรืออาจจะเป็นสแต็กของตัวดอสเอง ซึ่งขนาดของสแต็กอาจจะมีไม่เพียงพอสำหรับรูทีน-ป๊อบอัพจะทำงานได้ ดังนั้นจำเป็นต้องทำการเปลี่ยนสแต็กให้มาใช้งานสแต็กของ

โปรแกรมเรสซิเดนต์เอง โดยการเปลี่ยนค่าในรีจิสเตอร์สแต็ก (SS และ SP) จากการกระทำนี้จะเห็นโปรแกรมเรสซิเดนต์ที่จะสร้างโปรแกรมที่ไม่สามารถเรียกซ้ำ (Non-reentrant) ซึ่งเราจะแก้ไขปัญหานี้ได้โดยการสร้างแฟล็กขึ้นมาเพื่อใช้ตรวจสอบว่า กำลังทำงานอยู่ในรูทีนป๊อบอัพ โดยจะต้องกำหนดค่าแฟล็กนี้ (จะใช้ชื่อว่า running) ให้เป็น 1 ก่อนจะทำการเปลี่ยนค่าสแต็กและหลังจากทำงานในรูทีนป๊อบอัพเสร็จสิ้นและคืนค่าสแต็กเดิมเรียบร้อยแล้ว จึงจะกำหนดแฟล็ก running ให้เป็น 0

อินเตอร์รัพต์ของ Control-Break และ Control-C

ตามปกติการที่ผู้ใช้โปรแกรมกดคีย์ Ctrl-Break หรือ Ctrl-C จะให้โปรแกรมที่ทำงานอยู่ขณะนั้นเลิกการทำงาน ดังนั้นเพื่อป้องกันเหตุการณ์นี้เกิดกับโปรแกรมเรสซิเดนต์ก่อนจะเข้ารูทีนป๊อบอัพจะต้องเปลี่ยนอินเตอร์รัพต์ทั้งสองนี้ (อินเตอร์รัพต์ 01bh สำหรับ Control-Break และอินเตอร์รัพต์ 023h สำหรับ Control-C) โดยทำเปลี่ยนค่าในตารางอินเตอร์รัพต์เวกเตอร์ให้มาชี้ที่ฟังก์ชันต่าง ๆ ของโปรแกรมเรสซิเดนต์ เมื่อป๊อบอัพขึ้นจะไม่สนใจ Control-Break และ Control-C เมื่อออกจากรูทีนป๊อบอัพก็คืนค่าเวกเตอร์อินเตอร์รัพต์ทั้งสองคืน นอกจากนี้อีกสิ่งที่ควรจะทำก่อนเรียกรูทีนป๊อบอัพก็คือ การบอกดอสให้เลิกสนใจ Control-Break ซึ่งทำได้โดยใช้ฟังก์ชันดอสอินเตอร์รัพต์ 21h ฟังก์ชันที่ 33h แต่อย่างไรก็ตามต้องเก็บสถานะของดอสการตอบสนอง Control-Break เอาไว้เพื่อคืนให้แก่ดอสหลังจากออกจากรูทีนป๊อบอัพ ซึ่งในเทอร์โบซีได้เตรียมฟังก์ชันเอาไว้แล้วคือ getcbrk อ่านค่าการตอบสนอง Control-Break และ setcbrk กำหนดค่าการตอบสนอง Control-Break

อินเตอร์รัพต์ของ Critical Error

โปรแกรมที่ทำงานภายใต้ดอสเมื่อเกิดข้อบกพร่องในการติดต่ออุปกรณ์ต่าง ๆ เช่น ดิสก์ เครื่องพิมพ์ เป็นต้น ตัวอย่างของข้อบกพร่องได้แก่ การพยายามเขียนหรืออ่านดิสก์ขณะที่ฝาดิสก์ไดรฟ์ไม่ได้ปิด การพยายามพิมพ์ข้อความออกทางเครื่องพิมพ์แต่เครื่องพิมพ์ไม่พร้อมทำงาน เหตุการณ์เหล่านี้จะทำให้ดอสเรียกโปรแกรมอินเตอร์รัพต์ 24h คืออินเตอร์รัพต์ Critical Error ซึ่งการทำงานของอินเตอร์รัพต์ 24h นี้คือ จะพิมพ์ข้อความแสดงข้อผิดพลาดที่เกิดจากอุปกรณ์เหล่านั้น พร้อมทั้งรอคำตอบจากผู้ใช้โปรแกรมว่า จะยกเลิกการทำงานหรือจะพยายามทำงานนั้นอีกครั้ง หรือจะทำงานถัดไปต่อโดยไม่สนใจข้อผิดพลาดนั้น ๆ

Abort, Retry, Ignore?

ซึ่งคำตอบที่ได้รับจากผู้ใช้โปรแกรมนั้น อินเตอร์รัพต์ 24h จะส่งกลับไปยังดอสโดยผ่านทางรีจิสเตอร์ AL คำตอบคือ Abort ซึ่งถ้าเกิดขึ้นขณะโปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาทำงานอาจจะทำความเสียหายให้กับระบบได้ ดังนั้นเพื่อแก้ไขปัญหานี้ผู้เขียนโปรแกรมเรสซิเดนต์จะต้องจัดการแก้ไขอินเตอร์รัพต์ 24h เสียใหม่เอง แต่สำหรับในโปรแกรมตัวอย่างจะทำการเปลี่ยนแปลงอินเตอร์รัพต์ 24h ใหม่ โดยส่งคำตอบให้ดอสทำงานต่อไปโดยไม่สนใจข้อผิดพลาด (ซึ่งทำได้โดยการกำหนดค่า "ศูนย์" ให้รีจิสเตอร์ AL ก่อนออกจากอินเตอร์รัพต์ 24h)

Program Segment Prefix (PSP) และพื้นที่รับส่งข้อมูลกับดิสก์ (Disk Transfer Area, DTA)

ตามปกติก่อนดอสจะนำโปรแกรมลงหน่วยความจำเพื่อให้โปรแกรมทำงาน ดอสจะสร้างพื้นหน่วยความจำขนาด 256 ไบต์ เรียกว่า Program Segment Prefix (PSP) เอาไว้ส่วนบนสุดของโปรแกรม ซึ่งภายใน PSP จะมีข้อมูลสำหรับควบคุมการทำงานของโปรแกรมโดยเฉพาะที่สำคัญคือเรื่องดิสก์หรือไฟล์ โครงสร้างรายละเอียดของ PSP ให้ไว้ในตารางที่ 3 และก็เป็นเพราะว่าดอสนั้นจะควบคุมการทำงานของโปรแกรมได้เพียงหนึ่งโปรแกรม ซึ่งขณะที่ปลุกโปรแกรมเรสซิเดนต์ขึ้นมา ดอสก็ยังคงใช้ PSP ของโปรแกรมเดิมอยู่ ถ้าหากโปรแกรมเรสซิเดนต์มีการเขียนหรืออ่านไฟล์ก็อาจจะทำความเสียหายให้กับข้อมูลของโปรแกรมที่ทำอยู่ก่อนการปลุกโปรแกรมเรสซิเดนต์ได้ ดังนั้นในโปรแกรมเรสซิเดนต์ก่อนจะเรียกรูทีนป๊อบอัพ จึงควรจะบอกดอสให้เปลี่ยนการใช้ PSP จากของเดิมมาเป็น PSP ของโปรแกรมเรสซิเดนต์

ในพื้นที่ของดอสจะมีหน่วยความจำส่วนหนึ่งสำหรับเก็บตำแหน่งของ PSP ที่ดอสใช้ควบคุมโปรแกรมที่ทำงานอยู่ เช่นเดียวกับที่เก็บแฟล็ก Indos ที่ใช้บอกสถานะการทำงานดอสดังที่กล่าวแล้วในหัวข้อก่อน ในการเปลี่ยนแปลง PSP นั้นก็ทำโดยเปลี่ยนค่าในพื้นที่ส่วนนี้นั่นเอง ซึ่งดอสได้มีฟังก์ชันสำหรับใช้งานนี้อยู่ด้วยกัน 3 ฟังก์ชัน (แต่ไม่เปิดเผย) คือฟังก์ชัน 50h กำหนด PSP ใหม่ ฟังก์ชัน 51h อ่านค่า PSP และฟังก์ชัน 62h สำหรับการอ่านค่า PSP เช่นกัน ตารางที่ 4 เป็นตารางสรุปฟังก์ชันที่ไม่เปิดเผยและใช้งานโปร่แกรมเรสซิเดนต์

ฟังก์ชัน 50h และ 51h เริ่มมีตั้งแต่ดอสรุ่น 2.0 แต่ฟังก์ชัน 51h เมื่อใช้งานในอินเตอร์รัพต์ 28h พบว่ามีปัญหา (Bug) เกิดขึ้น เพราะฟังก์ชันนี้ไปใช้สแต็กร่วมกับฟังก์ชันอื่น อย่างไรก็ตามต่อมาดอสได้เพิ่มฟังก์ชัน 62h ขึ้นมาในดอสรุ่น 3.0 ซึ่งการทำงานจะเหมือนกับฟังก์ชัน 51h แต่ได้แก้ไขปัญหาที่เกิดขึ้นกับฟังก์ชัน 51h แต่ในโปรแกรมเรสซิเดนต์ในบทความนี้จะไม่ใช้ฟังก์-

ตารางที่ 3 ส่วนต่าง ๆ ของ PSP และความหมาย

เซตที่	ออฟเซ็ต	ขนาด (ไบต์)	ความหมาย
1	0000H-0001H	2	เป็นที่เก็บคำสั่ง INT 20H ซึ่งเป็นคำสั่งจบการทำงานของโปรแกรมเพื่อกลับไปสู่เอ็มเอสดอส
2	0002H-0003H	3	เซกเมนต์แอดเดรสของหน่วยความจำสูงสุด
3	0004H	1	สงวนไว้
4	0005H-0009H	5	เป็นคำสั่ง LONG CALL ไปยังฟังก์ชันของเอ็มเอสดอส เราสามารถใช้คำสั่ง JMP กระโดดมาที่แอดเดรสนี้ เพื่อเรียกใช้ฟังก์ชันของเอ็มเอสดอสได้
5	000AH-000DH	4	เก็บแอดเดรสของการเลิกงาน เมื่อโปรแกรมจบการทำงานด้วยคำสั่ง INT 20H หรือ INT 21H ด้วยฟังก์ชัน 00H, 31H หรือ 4CH
6	000EH-0011H	4	เก็บแอดเดรสที่โปรแกรมจะกระโดดไปทำงานเมื่อมีการกด CTRL-BREAK
7	00012H-0015H	4	เก็บแอดเดรสที่โปรแกรมจะกระโดดไปทำงานเมื่อเกิดข้อผิดพลาด ซึ่งส่วนใหญ่จะเป็นข้อผิดพลาดที่เกี่ยวข้องกับอุปกรณ์ไอโอ เช่น ดิสค์ไม่พร้อมทำงาน เนื่องจากไม่ได้ปิดฝา หรือเกี่ยวกับเครื่องพิมพ์ เช่นกระดาษหมด เป็นต้น
8	0016H-002BH	7	สงวนไว้
9	002CH-002DH	2	เป็นตัวชี้แอดเดรสของสภาพแวดล้อม (Environment) ว่าเก็บอยู่ที่ใด สภาพแวดล้อมนี้เป็นบริเวณที่เก็บข้อมูลต่าง ๆ เช่น เส้นทาง (PATH) ของการค้นหาไฟล์ หรือตัวแปรที่เกิดจากการใช้คำสั่ง SET หรือรูปแบบของสัญลักษณ์เตรียมพร้อมที่กำหนดด้วยคำสั่ง PROMPT เป็นต้น ข้อมูลสภาพแวดล้อมนี้จะเก็บอยู่ในรูป -ASCIIZ ซึ่งหมายถึงรหัสอัสกี้ที่ปิดท้ายด้วยศูนย์
10	002EH-005BH	45	สงวนไว้
11	005CH-006BH	15	เป็นที่เก็บบล็อกการควบคุมไฟล์ (File Control Block) ตัวที่ 1
12	006CH-007FH	19	เป็นที่เก็บบล็อกการควบคุมไฟล์ตัวที่ 2 จะสังเกตว่า FCB ตัวที่ 2 นี้จะซ้อนทับ FCB ตัวที่ 1 ในกรณีที่ใช้ FCB เพียงตัวเดียวจะไม่มีปัญหาใด แต่ถ้าต้องใช้ตัวที่ 2 ด้วย ต้องเก็บ FCB ตัวแรกก่อน
13	0080-00FFH	128	เป็นที่เก็บพารามิเตอร์ของบรรทัดคำสั่ง และใช้เป็นพื้นที่ส่งผ่านของดิสค์ (Disk Transfer Area) พื้นที่นี้จะเป็นบัฟเฟอร์หรือกันชนระหว่างดิสค์กับบริเวณที่เก็บข้อมูลจริง เมื่อมีการอ่านข้อมูลจากดิสค์สู่หน่วยความจำ ข้อมูลจะถูกอ่านลงมาที่นี่ก่อน แล้วจึงย้ายไปสู่บริเวณเก็บข้อมูลจริง ในกรณีการเขียนก็เช่นเดียวกัน

ตารางที่ 4 สรุปฟังก์ชันของดอส (Int 21h) ที่ไม่เปิดเผยและใช้งานในโปรแกรมเรสซิเดนต์

Function No.	Name of Function	Imput	Output	DOS version
034 h	Get INDOS Flag	AH = 034 h	ES = Segment of INDOS Flag BX = offset of INDOS Flag	2.0 and later
050 h	Set Current PSP	AH = 050 h BX = Segment of PSP to be made current	none	2.0 and later
051 h	Get PSP Segment	AH = 051 h	BX = Current PSP	2.0 and later
052 h	Get First MCB	AH = 052 h	ES : [BX – 2] point to First MCB	2.0 and later
062 h	Get PSP Segment	AH = 062 h	BX = Current PSP	3.0 and later

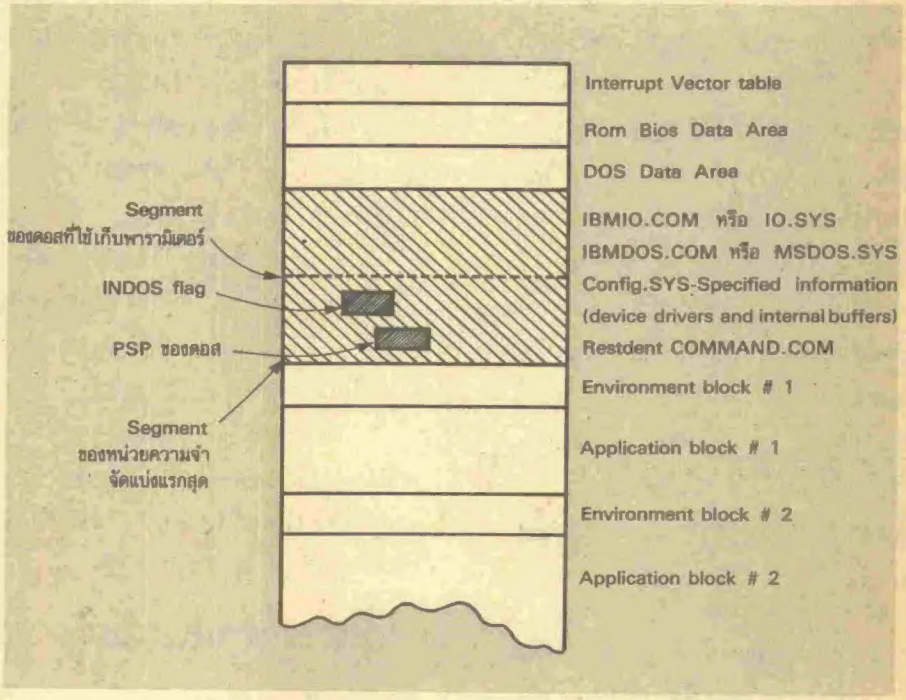
ชัน 62h ซึ่งเฉพาะในดอสรุ่น 3.0 ขึ้นไป เพราะจะทำให้โปรแกรมเรสซิเดนต์มีข้อจำกัด คือต้องใช้งานเฉพาะในดอสรุ่นที่ 3.0 ขึ้นไป เพื่อแก้ไขปัญหานี้ก็จะทำการค้นหาตำแหน่งและทำการเปลี่ยนค่าตรง ๆ ในพื้นที่หน่วยความจำของดอสที่ใช้เก็บ PSP โดยจะเริ่มต้นค้นจากจุดเริ่มต้นของดอส ซึ่งได้มาจากการค้นหาแฟล็ก InDOS (ในรีจิสเตอร์ ES ซึ่งให้เซกเมนต์เริ่มต้นของหน่วยความจำที่ดอสใช้งาน) จนกระทั่งถึงหน่วยความจำสุดท้ายของดอส ที่หาได้จากฟังก์ชัน 52h โดยที่ค่าซึ่งถูกชี้โดย EX : [BX-2] จะเป็นเซกเมนต์ของหน่วยความจำแรกสุดของการแบ่งหรือจัดสรรใช้งาน (รายละเอียดในเรื่อง "การจัดการหน่วยความจำของดอส" ไมโครฯ ฉบับ 50) ซึ่งมีอัลกอริทึมในการค้นหาเพื่อความแน่ใจว่าใช่พื้นที่เก็บ PSP ที่ถูกต้องโดยทำการเปรียบเทียบกับค่า PSP จริง ๆ ของโปรแกรมเรสซิเดนต์เมื่อโหลดโปรแกรมลงหน่วยความจำครั้งแรก เมื่อพบตำแหน่ง เพื่อความแน่ใจว่าไม่ใช่ค่าหน่วยความจำเท่ากันโดยบังเอิญ จึงลองเปลี่ยน PSP โดยใช้ฟังก์ชัน 50h ด้วยค่าอะไรก็ได้ที่รู้ค่าแล้วลองเปรียบค่าในตำแหน่งที่พบกับค่าที่ใช้เปลี่ยน ถ้าเท่ากันก็แสดงว่าพบตำแหน่งที่เก็บ PSP ของดอสแน่นอนแล้ว โดยจะนำค่าไปใช้ในการเปลี่ยน PSP ก่อนและหลังการป๊อบอัพของโปรแกรมเรสซิเดนต์

สำหรับพื้นที่รับส่งข้อมูลกับดิสก์ DTA นั้นปกติ (Default) แล้วจะอยู่เป็นส่วนหนึ่ง PSP ซึ่งสามารถจะกำหนดที่ใหม่ได้โดยจะต้องจองเนื้อที่จากดอสก่อน DTA นั้นจะใช้เป็นที่สำหรับที่พักข้อมูล (Buffer) ในการเขียนหรืออ่านดิสก์ และเป็นไปได้ที่โปรแกรมที่ทำงานอยู่ขณะที่โปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมานั้น อาจจะจองหรือสร้าง DTA ขึ้นมาใหม่สำหรับใช้งานเอง ดังนั้นเมื่อโปรแกรมเรสซิเดนต์ถูกปลุกขึ้นมาจึงต้องเก็บ DTA ของโปรแกรมที่ทำงานอยู่ก่อนเพื่อคืนให้กับดอสหลังจากที่รูทีนป๊อบอัพทำงานเสร็จ ฟังก์ชันสำหรับอ่านและกำหนด DTA อยู่ในตารางที่ 5 แต่สำหรับไลบรารี่ของเทอร์โบซีได้เตรียมฟังก์ชันไว้ให้แล้ว คือฟังก์ชัน getdta และ setdta

จอภาพและเคอร์เซอร์

เรื่องของจอภาพและเคอร์เซอร์กับโปรแกรมเรสซิเดนต์ก็เป็นส่วนหนึ่งที่สำคัญ ซึ่งการทำงานของโปรแกรมเรสซิเดนต์อาจจะต้องใช้พื้นที่บางส่วนของจอภาพ ดังนั้นก่อนจะให้โปรแกรมเรสซิเดนต์ขึ้นมาทำงานจะต้องเก็บสถานภาพของจอภาพเดิมเอาไว้ เพื่อคืนให้หลังโปรแกรมเรสซิเดนต์ทำงานเสร็จ โปรแกรมเรสซิเดนต์ที่ได้รับความนิยมในปัจจุบันสามารถจะเก็บจอภาพเดิมที่เป็นกราฟิก พร้อมทั้งคืนสภาพเดิมได้ถูกต้องหลังทำงานเสร็จ เช่น ไซด์คิกพลัส ซึ่งในเรื่องนี้ขอยกให้ผู้อ่านที่ต้องการเขียนโปรแกรมเรสซิเดนต์ไปศึกษาเอง

รูปที่ 1 แสดงที่อยู่ของ INDOS flag ซึ่งได้จากฟังก์ชัน 34h และ PSP ของดอส ซึ่งค้นหาโดยเริ่มจากเซกเมนต์ที่เก็บพารามิเตอร์ของดอสจนถึงหน่วยความจำจัดแบ่งอันแรกสุด



ตารางที่ 5 ฟังก์ชันของดอส (Int 21h) ในกำหนดและอ่านพื้นที่ DTA

Function No	Function Name	input	out put
1Ah	Set Disk Transfer	AH = 1Ah	None
	Address	DS : DX = Disk Transfer Address	
2Fh	Get Disk Transfer Address	AH = 2Fh	ES : BX Points to Disk Transfer Address

การป้องกันการโหลดโปรแกรมซ้ำ

การป้องกันการโหลดโปรแกรมเรสซิเดนต์ซ้ำสองทำได้การตรวจสอบสภาพโปรแกรมก่อนจะทำการติดตั้งโปรแกรม ซึ่งมีด้วยกัน 2 วิธีคือ

1. การค้นหาโปรแกรมในหน่วยความจำซึ่งดอสควบคุม คือค้นหาซึ่งตัวโปรแกรมจากการโหลดครั้งแรกใน Memory Control Block ของดอส

2. ในการโหลดโปรแกรมครั้งแรกนั้นโปรแกรมจะติดตั้งชื่อหรือสัญลักษณ์ของโปรแกรมเอาไว้ในเวกเตอร์อินเตอร์รัพต์ที่ว่างอยู่ (ไว้สำหรับโปรแกรมทั่วไปใช้งาน) ซึ่งได้แก่อินเตอร์รัพต์หมายเลข 60h ถึง 67h โดยการเลือกเอาอินเตอร์รัพต์ที่ว่าง จากนั้นในการโหลดโปรแกรมครั้งต่อไปก็จะทำการตรวจสอบกับอินเตอร์รัพต์หมายเลขต่าง ๆ ดังกล่าว ซึ่งถ้าพบก็จะไม่ติดตั้งโปรแกรม แต่จะเลิกโปรแกรมไปตามปกติ ทั้งสองวิธีนั้นวิธีแรกเป็นวิธีที่ถูกต้องที่สุด และวิธีที่สองนั้นง่ายและสะดวกต่อการติดตั้งและตรวจสอบ ซึ่งในโปรแกรมที่ให้เป็นตัวอย่างจะใช้วิธีที่สอง

การติดต่อส่งข้อมูลให้กับโปรแกรมเรสซิเดนต์ที่ฝังโปรแกรมไปแล้ว

การส่งผ่านข้อมูลหรือพารามิเตอร์ให้กับโปรแกรมเรสซิเดนต์ที่ฝังโปรแกรมไปแล้วนั้น ทำได้โดยการโหลดหรือเรียกใช้งานโปรแกรมเรสซิเดนต์อีกครั้งหนึ่ง ซึ่งส่งผ่านข้อมูลทางรีจิสเตอร์ ผ่านทางอินเตอร์รัพต์ (60h ถึง 67h) ซึ่งโปรแกรมเรสซิเดนต์ที่โหลดก่อนได้ทำการเปลี่ยนแปลงให้เป็นรูทีนสำหรับควบคุมการทำงานของโปรแกรม ภายในรูทีนนี้จะทำการตรวจสอบค่ารีจิสเตอร์ที่ได้รับมาว่าจะให้ทำงานอะไร (คล้าย ๆ ฟังก์ชันบริการของไบออสหรือดอสนั่นเอง)

การถอนโปรแกรมเรสซิเดนต์ออกจากหน่วยความจำ

ก่อนถอนโปรแกรมออกจากหน่วยความจำ โปรแกรมเรสซิเดนต์จะทำการตรวจสอบก่อนว่ามีโปรแกรมหรือโปรแกรมเรสซิเดนต์ใดที่ใช้งานอินเตอร์รัพต์ที่ตรงกับของโปรแกรมเรสซิเดนต์ ซึ่งทำได้โดยเปรียบเทียบอินเตอร์รัพต์เวกเตอร์เดิมที่เก็บไว้ ถ้าไม่ตรงกันนั่นหมายถึงมีโปรแกรมหรือโปรแกรมเรสซิเดนต์ตัวอื่นโหลดอยู่ทีหลัง โปรแกรมจะไม่ยอมถอนออกจากหน่วยความจำ จนกว่าจะถอนโปรแกรมหรือโปรแกรมเรสซิเดนต์อื่นออกก่อน ทั้งนี้เพื่อจะได้อินเตอร์รัพต์ที่ถูกต้อง เป็นการป้องกันการทำงานของระบบด้วย ขั้นตอนการทำงานมีดังนี้

1. ถ้าในโปรแกรมเรสซิเดนต์มีการเปิดใช้ไฟล์จะต้องทำการปิดไฟล์เสียก่อน

2. คืนอินเตอร์รัพต์เวกเตอร์เดิมที่เก็บไว้คืนแก่ระบบ

3. De-allocate หน่วยความจำ ซึ่งจะได้แก่ Environment Block และ Application Block (คือตัวโปรแกรมส่วน Enviroment Block นั้นหาได้จากค่า PSP)

กฎและข้อหลีกเลี่ยงสำหรับการเขียนโปรแกรมด้วยเทอร์โบซี

ในเทอร์โบซีนั้นก่อนจะทำงานที่ฟังก์ชัน main นั้นจะทำงานส่วน Start-up Code ของเทอร์โบซีก่อน ซึ่งในส่วนนี้มีการเปลี่ยนแปลงอินเตอร์รัพต์เวกเตอร์บางอย่าง ซึ่งเมื่อจบฟังก์ชัน main จึงจะคืนค่าเดิมให้ แต่ในโปรแกรมเรสซิเดนต์นั้นไม่ได้จบโปรแกรมตามปกติ ดังนั้น อินเตอร์รัพต์บางตัวอาจจะไม่ได้คืน ซึ่งจะทำให้ระบบเสียหายได้ ตัวอย่างเช่น อินเตอร์รัพต์หมายเลขศูนย์ (Divide-by-Zero) ได้ถูกเปลี่ยนก่อนเรียกฟังก์ชัน main ซึ่งก่อนจะใช้ฟังก์ชัน Keep จะต้องคืนค่านี้ให้ถูกต้องก่อน และเนื่องจากเทอร์โบซีรุ่น 1.5 ขึ้นไปเก็บค่าอินเตอร์รัพต์เวกเตอร์ไว้ที่ตัวแปร Zero DivVector ดังนั้นเราจึงสามารถคืนอินเตอร์รัพต์ได้อย่างถูกต้อง ปัญหาอีกอันหนึ่งก็คือเรื่องไลบรารี่คณิตศาสตร์ (floating-point math library) ซึ่งจะใช้อินเตอร์รัพต์บางหมายเลข และทำการเปลี่ยนแปลงค่าอินเตอร์รัพต์เวกเตอร์ใน Start-up Code เช่นกัน ดังนั้นไม่ควรจะใช้ฟังก์ชันทางคณิตศาสตร์ (floating point) ในโปรแกรมเรสซิเดนต์ และที่สำคัญจะลืมไม่ได้คือ อย่าใช้ฟังก์ชันในไลบรารี่ที่ใช้ฟังก์ชันของดอส คือฟังก์ชัน 0 ถึง 0ch การอ่านคีย์บอร์ดเรียกฟังก์ชันบริการของไบออสแทน ซึ่งเขียนไว้ให้เป็นฟังก์ชัน get_char

โปรแกรมและการใช้งาน

สำหรับโปรแกรมที่ให้ในฉบับนี้จะเป็นโปรแกรมสำหรับเทอร์โบซีก่อน ซึ่งแบ่งออกเป็น 3 ไฟล์ ซึ่งไฟล์ที่ 1 และ 2 จะเป็นส่วนสำคัญควบคุมโปรแกรม ส่วนไฟล์ที่ 3 เป็นตัวอย่างการทำงานของป๊อบอัพรูทีน โดยส่วนนี้ผู้อ่านอาจจะเขียนขึ้นมาเองก็ได้ จะให้ทำงานใดตามความประสงค์ แต่สำหรับตัวอย่างเมื่อป๊อบอัพขึ้นมาจะแสดงการใช้หน่วยความจำของดอสว่ามีโปรแกรมใด ๆ ค้างอยู่ในหน่วยความจำบ้าง (สามารถนำมาใช้ตรวจสอบไวรัสได้) การเปลี่ยนคีย์พิเศษสำหรับป๊อบอัพจะทำที่ไฟล์ที่ 1 รวมทั้งชื่อหรือสัญลักษณ์โปรแกรมด้วย ในฉบับหน้าจะเป็นโปรแกรมเรสซิเดนต์สำหรับไมโครซอฟต์ซี สำหรับฉบับนี้คงต้องจบลงแค่นี้ครับ

รูปที่ 2 ไฟล์ Header ของไฟล์ TSR.C

```
/* ------  tsr.h ------- */

void resinit(void);
void resident_psp(void);
void interrupted_psp(void);
unsigned resident(char *,void interrupt(*)());
void terminate();
void restart(void);
void suspend(void);
int get_char(void);
void popup(void);
void openfiles(void);
void closefiles(void);
```

รูปที่ 3 ไฟล์ TSR.C เปลี่ยนแปลงคีย์พิเศษและสัญลักษณ์โปรแกรม

```
#include <dos.h>
#include <stdio.h>
#include <string.h>
#include <dir.h>
#include "tsr.h"

static union REGS rg;
/* -----  Modify for the specific TSR  ----- */
extern unsigned _heaplen = 512;          /* HEAP SIZE */
extern unsigned _stklen =512;            /* STACK SIZE */
unsigned scancode = 42; /* Left Shift */ /* HOT KEY SCANCODE */
unsigned keymask = 8;         /* Alt */  /* HOT KEY STATUS MASK */
char signature [] = "Memory Control Block";

static void interrupt ifunc();

main(argc, argv)
int argc;
char *argv[];
{
       int ivec;
       if ((ivec = resident(signature, ifunc)) != 0) {
          rg.x.ax = 0;
          if (strcmp(argv[1], "quit") == 0)
             rg.x.ax = 1;
          else if (strcmp(argv[1], "restart") == 0)
             rg.x.ax = 2;
          else if (strcmp(argv[1], "wait") == 0)
             rg.x.ax = 3;
         if (rg.x.ax) {
           int86(ivec, &rg, & rg);
           return;
         }
       }
       else {
           printf("\nResident MCB (Memory control block) is loaded");
           openfiles();
           resinit();
      }
}
void interrupt ifunc(bp, di, si, ds, es, dx, cx, bx, ax)
{
      if (ax == 1)           /* quit */
         terminate();
      else if (ax == 2)      /* restart */
         restart();
      else if (ax == 3)      /* wait */
         suspend();
}
```

รูปที่ 4 ไฟล์ Resident.C รวบรวมฟังก์ชันช่วยในการเรสซิเดนต์

```
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include "tsr.h"

#define KEYBD   9
#define TIMER   0x1c
#define DISK    0x13
#define VIDEO   0x10
#define ZERODIV 0
#define INT28   0x28
#define CRIT    0x24
#define CTRLC   0x23
#define CTRLBRK 0x1b

static void interrupt (*oldbreak)();
static void interrupt (*oldctrlc)();
static void interrupt (*oldtimer)();
static void interrupt (*old28)();
static void interrupt (*oldkb)();
static void interrupt (*olddisk)();
static void interrupt (*oldcrit)();
static void interrupt (*ZeroDivVector)();

static void interrupt newtimer();
static void interrupt new28();
static void interrupt newkb();
static void interrupt newdisk();
static void interrupt newcrit();
static void interrupt newbreak();

static union REGS rg;
static struct SREGS seg;

static unsigned sizeprogram;
static unsigned dosseg;
static unsigned dosbusy;
static char far *mydta;
static unsigned myss;
static unsigned stack;
static unsigned intpsp;
static unsigned psps[2];
static int pspctr;
static int resoff;
static int running;
static int popflg;
static int diskflag;
static int cflag;

static void restorm(void);
static void pspaddr(void);
static void dores(void);

void resinit()
{
       segread(&seg);
       myss = seg.ss;
       stack = _SP;
       rg.h.ah = 0x34;
       intdos(&rg, &rg);
       dosseg = _ES;
```

รูปที่ 4 (ต่อ)

```
        dosbusy = rg.x.bx;
        mydta = getdta();
        pspaddr();
        oldtimer = getvect(TIMER);
        old28 = getvect(INT28);
        oldkb = getvect(KEYBD);
        olddisk = getvect(DISK);
        setvect(TIMER, newtimer);
        setvect(KEYBD, newkb);
        setvect(INT28, new28);
        setvect(DISK, newdisk);
        sizeprogram = myss + ((stack + 50) / 16) - _psp;
        setvect(ZERODIV, ZeroDivVector);
        keep(0, sizeprogram);
}


static void interrupt newbreak()
{
        return;
}

void interrupt newcrit(bp, di, si, ds, es, dx, cx, bx, ax, ip, cs, flgs)
{
        ax = 0;
        cflag = flgs;
}

void interrupt newdisk(bp, di, si, ds, es, dx, cx, bx, ax, ip, cs, flgs)
{
        diskflag++;
        (*olddisk)();
        ax = _AX;
        cx = _CX;
        dx = _DX;
        newcrit();
        flgs = cflag;
        --diskflag;
}

static int kbval;
void interrupt newkb()
{
        extern unsigned scancode;
        extern unsigned keymask;

        if (scancode && inportb(0x60) == scancode) {
           kbval = peekb(0, 0x417);
           if (!resoff && ((kbval & keymask) << keymask) != 0) {
              kbval = inportb(0x61);
              outportb(0x61, kbval | 0x80);
              outportb(0x61, kbval);
              disable();
              outportb(0x20, 0x20);
              enable();
              if (!running)
                 popflg = 1;
              return;
           }
        }
        (*oldkb)();
}
```

```
void interrupt newtimer()
{
        extern unsigned scancode;
        extern unsigned keymask;
        (*oldtimer)();
        if (!scancode) {
           kbval = peekb(0, 0x417);
           if (!resoff && ((kbval & keymask) << keymask) != 0)
              if (!running)
                 popflg = 1;
        }
        if (popflg && peekb(dosseg, dosbusy) == 0)
           if (diskflag == 0) {
              outportb(0x20, 0x20);
              popflg = 0;
              dores();
           }
}

void interrupt new28()
{
        (*old28)();
        if (popflg && peekb(dosseg, dosbusy) != 0) {
           popflg = 0;
           dores();
        }
}

void resident_psp()
{
        int pp;
        intpsp = peek(dosseg, *psps);
        for (pp = 0; pp < pspctr; pp++)
            poke(dosseg, psps[pp], _psp);
}

void interrupted_psp()
{
        int pp;
        for (pp = 0; pp < pspctr; pp++)
            poke(dosseg, psps[pp], intpsp);
}

static void dores()
{
        static char far *intdta;
        static unsigned intsp;
        static unsigned intss;
        static unsigned ctrl_break;
        running = 1;
        disable();
        intsp = _SP;
        intss = _SS;
        _SP = stack;
        _SS = myss;
        enable();
        oldcrit = getvect(CRIT);
        oldbreak = getvect(CTRLBRK);
        oldctrlc = getvect(CTRLC);
        setvect(CRIT, newcrit);
        setvect(CTRLBRK, newbreak);
        setvect(CTRLC, newbreak);
```

รูปที่ 4 (ต่อ)

```
        ctrl_break = getcbrk();
        setcbrk(0);
        intdta = getdta();
        setdta(mydta);
        resident_psp();
        popup();
        interrupted_psp();
        setdta(intdta);
        setvect(CRIT, oldcrit);
        setvect(CTRLBRK, oldbreak);
        setvect(CTRLC, oldctrlc);
        setcbrk(ctrl_break);
        disable();
        _SP = intsp;
        _SS = intss;
        enable();
        running = 0;
}

static int avec = 0;

unsigned resident(signature, ifunc)
char *signature;
void interrupt(*ifunc)();
{
        char *sg;
        unsigned df;
        int vec;

        segread(&seg);
        df = seg.ds - seg.cs;
        for (vec = 0x60; vec < 0x68; vec++) {
            if (getvect(vec) == NULL) {
              if (!avec)
                avec = vec;
              continue;
            }
            for (sg = signature; *sg; sg++)
                if (*sg != peekb(peek(0, 2 + vec * 4) + df, (unsigned)sg))
                  break;
            if (!*sg)
              return vec;
        }
        if (avec)
            setvect(avec, ifunc);
        return 0;
}

static void pspaddr()
{
        unsigned adr = 0;
        unsigned enddos;
        rg.h.ah = 0x52;
        intdos(&rg, &rg);
        enddos = _ES;
        enddos = peek(enddos, rg.x.bx - 2);
        while (pspctr < 2 &&
                    (unsigned)((dosseg << 4) + adr) < (enddos << 4)) {
            if (peek(dosseg, adr) == _psp) {
              rg.h.ah = 0x50;
              rg.x.bx = _psp + 1;
              intdos(&rg, &rg);
              if (peek(dosseg, adr) == _psp + 1)
                  psps[pspctr++] = adr;
              rg.h.ah = 0x50;
              rg.x.bx = _psp;
              intdos(&rg, &rg);
            }
            adr++;
        }
}

static void resterm()
{
        closefiles();
        setvect(TIMER, oldtimer);
        setvect(KEYBD, oldkb);
        setvect(INT28, old28);
        setvect(DISK, olddisk);
        setvect(avec, (void interrupt(*)()) 0);
        freemem(peek(_psp, 0x2c));
        freemem(_psp);
}

void terminate()
{
      if (getvect(DISK) == (void interrupt(*)()) newdisk)
          if (getvect(KEYBD) == newkb)
              if (getvect(INT28) == new28)
                  if (getvect(TIMER) == newtimer) {
                      resterm();
                      return;
                  }
      resoff = 1;
}

void restart()
{
        resoff = 0;
}

void suspend()
{
        resoff = 0;
}

int get_char()
{
        int c;

        while (1) {
              rg.h.ah = 1;
              int86(0x16, &rg, &rg);
              if (rg.x.flags & 0x40) {
                  int86(0x28, &rg, &rg);
                  continue;
              }
              rg.h.ah = 0;
              int86(0x16, &rg, &rg);
              if (rg.h.al == 0)
                  c = rg.h.ah | 128;
              else
                  c = rg.h.ah;
              break;
        }
}
```

รูปที่ 5 ตัวอย่างโปรแกรมป๊อบอัพ ซึ่งจะแสดงการใช้หน่วยความจำว่ามีโปรแกรมใดบ้างอยู่ในหน่วยความจำ

```
#include <stdio.h>
#include <dos.h>
#include <conio.h>
#include "tsr.h"

#define COL     1
#define MAXROW 17

/*--------- Golbal Variable ----------*/
int    row = 5 ;
static int cnt;
static int mcbnum;
unsigned char ccnum;
unsigned prev_pid;
typedef unsigned ALINE [80];
ALINE  bufferSCRN [25];

/*--------- Golbal Declaration ----------*/
struct MCB
{
        char chain;
        unsigned pid;
        unsigned psize;
        char unused [11];
} ;

typedef struct MCB huge *PTRMCB;

/*--------- Function Prototype ----------*/
void ClearWindow ();
void far * FindFirstMCB (void);
void PrintHeader (void);
void PrintMCB (PTRMCB);
void PrintPidOwner (unsigned pid,unsigned parent);

/*----------- Function Popup -------------*/
void popup()
{
        PTRMCB ptrmcb;
        int oldX, oldY;

        oldX = wherex ();
        oldY = wherey ();
        textattr (0x78);
        gettext(1,1,80,25,bufferSCRN);
        cnt = ccnum = 0;
        mcbnum = 1;
        prev_pid = 0xFFFF;
        ptrmcb = FindFirstMCB ();
        do
          {
          ClearWindow ();
          row = 5;
          gotoxy (COL,row);
          PrintHeader ();
          do
            {
            PrintMCB (ptrmcb);
            ptrmcb += ptrmcb->psize + 1;
            } while ( (row <= MAXROW) && (ptrmcb->chain == 'M') );
          PrintMCB (ptrmcb);
          gotoxy (COL,row++);
          cprintf ("-----------------------------------------------");
          cprintf ("-----------------\n");
          if ((row > MAXROW) && (ptrmcb->chain == 'M'))
            {
            gotoxy (COL,row++);
            cprintf ("More. . ...");
            get_char();
            }
          } while (ptrmcb->chain == 'M');
        gotoxy (COL,row++);
        cprintf ("End. . ...");
        get_char();
        puttext(1,1,80,25,bufferSCRN);
        gotoxy (oldX,oldY);
}

void ClearWindow ()
{
        int row;

        for (row = 5; row < MAXROW +3 ; row++)
            {
            gotoxy (COL,row);
            cprintf ("                                                  ");
            cprintf ("                    ");
            }
}

/*---------- Find First Memory Control Block ---------*/
void far * FindFirstMCB (void)
{
        union REGS regs;
        struct SREGS sregs;
        unsigned far *segmentptr;

        regs.h.ah = 0x52;
        intdosx(&regs,&regs,&sregs);
        segmentptr = MK_FP (sregs.es,regs.x.bx - 2);
        return (MK_FP (*segmentptr, 0));
}

/*---------- Print out Header For Out Put ------------*/
void PrintHeader (void)
{
        gotoxy (COL,row++);
        cprintf ("-----------------------------------------------");
        cprintf ("-----------------\n");
        gotoxy (COL,row++);
        cprintf ("MCB  MCB ID PID     MCB    PAR-ENV  OWNER\n");
        gotoxy (COL,row++);
        cprintf ("No. SEG           SIZE  ID   BLK?\n");
        gotoxy (COL,row++);
        cprintf ("-----------------------------------------------");
        cprintf ("-----------------\n");
}

/*-------- Print Out the Information of MCB Pointer ---------*/
void PrintMCB (PTRMCB pm)
{

        unsigned parid;
        unsigned mcbseg;
        char envf;
        unsigned envseg;

        parid = * (unsigned far *)MK_FP (pm->pid,0x16);
        mcbseg = FP_SEG (pm);
        envseg = * (unsigned far *) MK FP (pm->pid,0x2c);
```

→

รูปที่ 5 (ต่อ)

```
        envf = (mcbseg + 1 == envseg) ? 'Y' : 'N';
        if (parid == pm->pid)
           cnt++;
        if (!envseg && cnt == 2)
           envf = 'Y';
        gotoxy (COL,row++);
        cprintf ("%2.2u%06.4X%2.1c%06.4X%7.1u",
                mcbnum++, mcbseg, pm->chain, pm->pid,
                (long) pm->psize*16);
        cprintf ("%5.4X %-5.1c", parid, envf);
        PrintPidOwner (pm->pid,parid);
}

/*------- Print Out Owner MCB String -------*/
void PrintPidOwner (unsigned pid, unsigned parent)
{
        unsigned far *envsegptr;
        char far *envptr;
        unsigned far *envsizeptr;
        unsigned envsize;

        switch (pid)
                {
                case 0 : cprintf ("FREE MEMORY CONTROL BLOCK\n");
                         return;
                case 8 : cprintf ("IBMDOS.COM/MSDOS.SYS\n");
                         return;
                }
        envsegptr = (unsigned  far *) MK_FP (pid,0x2c);
        envptr = (char far *) MK_FP (*envsegptr,0);
        envsizeptr = (unsigned far *) MK_FP (*envsegptr - 1,0x3);
        envsize = *envsizeptr * 16;
        if (pid == parent)
                {
                if (prev_pid != pid)
                   ccnum++;
                cprintf ("COMMAND.COM COPY #%-2u\n",ccnum);
                prev_pid = pid;
                return;
                }
        while (envsize)
                {
                while (--envsize && *envptr++);
                if (!*envptr && *(unsigned far *)(envptr+1) == 0x1)
                   {
                   envptr += 3;
                   break;
                   }
                }
        if (envsize)
                {
                while (*envptr)
                  cprintf ("%c",*envptr++);
                cprintf ("\n");
                }
        else
          cprintf ("UNKOWN OWNER\n");
}

void openfiles()
{
}

void closefiles()
{
}
```


AI-EN INTERNATIONAL CO.,LTD.

มีความต้องการรับสมัครพนักงานตังต่อไปนี้

1. พนักงานวิจัยและพัฒนาผลิตภัณฑ์ทางด้านคอมพิวเตอร์

คุณสมบัติ - วุฒิปริญญาตรีวิศวกรรมคอมพิวเตอร์ หรือเทียบเท่า
- อายุไม่เกิน 30 ปี
- หากมีประสบการณ์จะพิจารณาเป็นพิเศษ

2. ช่างเทคนิคคอมพิวเตอร์

คุณสมบัติ - วุฒิ ป.ว.ส.อิเล็กทรอนิคส์ หรือเทียบเท่า
- อายุไม่เกิน 30 ปี
- มีความรู้ความเข้าใจในการใช้งานและตรวจซ่อมอุปกรณ์ประเภท IBM PC เป็นอย่างดี
- มีใบขับขี่รถจักรยานยนต์และรถยนต์
- หากมีประสบการณ์จะพิจารณาเป็นพิเศษ

3. SYSTEM ANALYSIS

คุณสมบัติ - วุฒิปริญญาตรีสาขาคอมพิวเตอร์ หรือเทียบเท่า อายุไม่เกิน 30 ปี
- สามารถเขียนภาษา C, PASCAL, dBASE, ASSEMBLY
- มีความเข้าใจระบบ ACCOUNTING SYSTEM, INVENTORY SYSTEM
- ต้องมีประสบการณ์การเขียนโปรแกรมอย่างน้อย 1 ปี

สนใจติดต่อและสมัครได้ที่:

บริษัท ไอ เอน อินเตอร์เนชั่นแนล จำกัด
1413/3-5 ถนนสุขุมวิท ใกล้ ร.พ.สุขุมวิท
พระโขนง กรุงเทพฯ 10110 โทร : 392-4476-8